

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

5

## APPLICATION FOR LETTERS PATENT

10

**FOR**

# MULTIPLE CRYPTOGRAPHIC KEY LINKING SCHEME ON A COMPUTER SYSTEM

15

Assignee: **Compaq Computer Corporation**

Assignee Ref. No.: P00-3017

Express Mail No. EL655031525US

## MULTIPLE CRYPTOGRAPHIC KEY LINKING SCHEME ON A COMPUTER SYSTEM

### 5 RELATED APPLICATIONS

This application is related to and incorporates herein by reference U.S. Applications entitled as follows:

“Scalable Computer System Using Password-Based Private Key Encryption” (Docket No. 20206.30 (P003014)), Serial No. \_\_\_\_\_, Filed \_\_\_\_\_;

10 “Method And Apparatus For Enforcing The Separation Of Computer Operations And Business Management Roles In A Cryptographic System” (Docket No. 20206.31 (P003015)), Serial No. \_\_\_\_\_, Filed \_\_\_\_\_;

“Software Process Pre-Authorization Scheme For Applications On A Computer System” (Docket No. 20206-32 (P00-3016)), Serial No. \_\_\_\_\_, Filed \_\_\_\_\_;

15 “Centralized Cryptographic Key Administration Scheme For Enabling Secure Context-Free Application Operation” (Docket No. 20206-34 (P00-3416)), Serial No. \_\_\_\_\_, Filed \_\_\_\_\_;

“Scalable Computer System Using Remote Agents To Manipulate Cryptographic Keys” (Docket No. 20206-35 (P00-3417)), Serial No. \_\_\_\_\_, Filed \_\_\_\_\_;

20 “Method And Apparatus For Cryptographic Key Rollover During Operation” (Docket No. 20206-36 (P00-3418)), Serial No. \_\_\_\_\_, Filed \_\_\_\_\_;

“Computer System Having An Autonomous Process For Centralized Cryptographic Key administration” (Docket No. 20206-37 (P00-3019)), Serial No. \_\_\_\_\_, Filed \_\_\_\_\_  
\_\_\_\_\_; and

25 “Computer System Employing A Split-Secret Cryptographic Key Linked To A Password-Based Cryptographic Key Security Scheme” (Docket No. 20206-38 (P00-3420)), Serial No. \_\_\_\_\_, Filed \_\_\_\_\_.

### BACKGROUND OF THE INVENTION

#### 30 Field of the Invention

The present invention is related to computer security. More specifically, the present invention is related to cryptographic systems on computer servers.

Description of the Related Art

E-Commerce

The advent of the Internet has spawned a new means for conducting business.

5 Commerce that is conducted online, such as via the Internet or virtual private networks, is called e-commerce. E-commerce mimics many of the steps of regular commerce. However, because of the nature of cyberspace, the parties may never have met and may never meet. To accommodate business transactions where the parties never meet or know each other, various schemes have been devised to ensure secure and verifiable business transactions.

10 The security of the e-commerce transaction is accomplished through encryption schemes. The authenticity and other necessary aspects of commerce are handled through trust relationships. Often, these trust relationships are implemented through trusted third parties and are erected as part of the online business infrastructure.

15 Computing systems evolved away from mainframe computers in the 1960's and 1970's to a distributed environment consisting mainly of personal computers in the 1980's and early 1990's. However, with the advent of the Internet, powerful servers (descendants of the old mainframe computers) have regained their former importance. Internet-connected servers now run software applications for client systems and perform business-to-business transactions and business-to-consumer transactions. In many cases, these transactions include sensitive information, which must be protected against unwanted exposure (privacy) or modification (integrity), or both. In some cases, there's a requirement that there be strong evidence of an event having taken place (non-repudiation), to further the resolution of disputes. People working in the field of cryptography have developed various schemes and methods have to provide such facilities. One of the crucial infrastructures of online business is trust. A trust relationship can 20 be erected using computer servers (and clients) that are equipped with software encryption applications.

25 A typical prior art public key encryption scheme is the RSA scheme, which is described in U.S. Patent No. 4,405,829 to Rivest et al; R.L. Rivest, A. Shamir, and L.M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, v. 21, n. 2, February 1978, pp. 120-126; "R.L. Rivest, A. Shamir, and L.M. Adleman, "On Digital Signatures and Public Key Cryptosystems," MIT Laboratory for Computer Science,

Technical Report, MIT/LCS/TR-212, January, 1979. The RSA scheme is used both for encryption and digital signatures. The RSA scheme is often combined with other technologies to provide privacy, integrity, and non-repudiation. Cryptographic systems, and the terminology used in the discipline, are described in "Applied Cryptography" by Bruce Schneier (John Wiley & Sons, Inc., New York, 1996).

### RSA Scheme

According to Bruce Schneier, "RSA gets its security from the difficulty of factoring large numbers. The public and private keys are functions of a pair of large prime numbers (100 to 200 digits or even larger). Recovering the plaintext from the public key and the ciphertext conjectured to be equivalent to factoring the product of the two primes." Bruce Schneier, "Applied Cryptography" Second Edition, John Wiley & Sons, Inc., New York, 1996, pp. 467.

Under the RSA scheme, to generate the two keys, one chooses two random large prime numbers,  $p$  and  $q$ . For maximum security, one chooses both  $p$  and  $q$  to be equal in length. Next, one computes the product:

$$n = pq$$

Then randomly choose the encryption key,  $e$ , such that  $e$  and  $(p - 1)(q - 1)$  are relatively prime. Finally, one uses the extended Euclidean algorithm to compute the decryption key,  $d$ , such that

$$ed = 1 \bmod (p - 1)(q - 1)$$

or, upon solving for the decryption key  $d$ ,

$$ed = e^{-1} \bmod (p - 1)(q - 1)$$

From the equations, it is clear that both  $d$  and  $n$  are relatively prime. The numbers  $e$  and  $n$  are the public key; the number  $d$  is the private key. The two primes,  $p$  and  $q$  are no longer needed and are discarded immediately – never to be disclosed or revealed.

Again, referring to Schneier, to encrypt a message, you first divide the message into numerical blocks smaller than  $n$  (with binary data, one simply chooses the largest power of 2 less than  $n$ ). For example, if both  $p$  and  $q$  are 100-digit primes, then  $n$  will have just under 200 digits and each message block,  $m_i$ , should be just under 200 digits long. The encrypted message,  $c$ , will be made up of similarly sized message blocks,  $c_i$ , of about the same length. The encryption formula is simply:

$$c_i = m_i^e \bmod n$$

To decrypt a message, one takes each encrypted block,  $c_i$ , and computes:

$$m_i = c_i^d \bmod n$$

Bloom-Shamir

5 There is another prior-art scheme that deals with the problem of splitting a secret into several components so that no one individual or group of individuals can produce the secret unless the required number of components are available. Often called an M-out-of-N scheme, it allows the customer to reduce the risk of malfeasance by requiring that M out of N (where M is one or more but less than N) people all agree to certain acts. One such M-out-of-N 10 implementing algorithm is the Bloom-Shamir algorithm that is defined in "Generalized Linear Threshold Scheme" by S.C. Kothari, (Proceedings of CRYPTO 84). See also: S. C. Kothari, "Generalized Linear Threshold Scheme," in Advances in Cryptology -- CRYPTO '84", G. R. Blakley and D. Chaum, eds.; and Lecture Notes in Computer Science volume 196 (1985), pages 231-241.

15 Software tools are available for implementing cryptographic schemes into software applications and user-interfaces. One such tool kit is called "BSAFE" and is produced by RSA Security of Bedford, Massachusetts. Information regarding BSAFE and other products, and about public key infrastructure, is available at the RSA web site at <http://www.rsasecurity.com/>.

20 Public Key Cryptographic Standards (PKCS)

25 Widely used methods of performing cryptographic operations are described in the Public Key Cryptographic Standards (PKCS), a set of standards for public-key cryptography developed by RSA Laboratories [PKCS]. The present invention makes use of:

PKCS#1 – a mechanism for encrypting and signing data; and

PKCS#5 – password-based cryptography method.

Authenticode

30 The Microsoft Corporation of Redmond, Washington, developed a technology in 1996 that enables users of the Internet's World Wide Web to download binary code (libraries and programs) in a manner that ensures the authenticity of the code. This technology has been dubbed "Authenticode" and is the subject of a white paper published by Microsoft in 1996

entitled "Microsoft Authenticode Technology, Ensuring Accountability and Authenticity for Software Components on the Internet." The document is available via the Internet at: <http://msdn.microsoft.com/workshop/security/authcode/authwp.asp> and is herein incorporated by reference.

5 According to the white paper,

10 "...[u]sing Microsoft Authenticode technology, end users can be assured of accountability and authenticity for software components they download over the Internet. Authenticode alerts users before Web sites download executable files to their computers. If code is signed, Authenticode presents the certificate so the user knows that the code hasn't been tampered with and so the user can see the code's publisher and the certificate authority. Based on their experience with and trust in the software publisher, users can decide what code to download on a case-by-case basis."

15 "Digital certificates are issued by independent certificate authorities such as VeriSign to commercial and individual software publishers. The certificate authority verifies the identity of each person or company registering, assuring that those who sign their code can be held accountable for what they publish. After successfully completing the verification process, the certificate authority issues the software publishing certificate to the publisher, who then signs its code before shipping an application."

20 "Users benefit from this software accountability because they know who published the software and that the code hasn't been tampered with. In the extreme and remote case that software performs unacceptable or malicious activity on their computers, they can also pursue recourse against the publisher. This accountability and potential recourse serve as a strong deterrent to the distribution of harmful code."

25 "Developers and Webmasters benefit tremendously from Authenticode as well. By signing their code, developers build a trusted relationship with users, who can learn to confidently download software from that publisher or Web site. Moreover, end users can make educated decisions about what software to download, knowing who published the software and that it hasn't been tampered with." Authenticode white paper, page 1.

### The Operational Paradigms

30 The growth of e-commerce requires that end-users be assured that their transactions are private, unmodified, and provable. The cryptographic techniques described above are often used to provide these attributes. In such a transaction, it is important to note the role of the originator of the transaction in exercising independent will to do the transaction. In the workstation

paradigm, it is possible to follow the steps of a transaction from the originator's perspective as follows, assuming that the originator is using, for example, a personal computer or terminal:

The originator decides what is to be done. This is the act of independent will, and will most often be done by a live person. Using the computer, the originator supplies the necessary 5 information defining the nature of the transaction. This could be something like "Please transfer \$100.00 from my checking account to my savings account." How this is done is not really important. The originator instructs the computer to sign and encrypt the data. The encryption step will ensure the privacy of the information as it flows from the computer to its destination (perhaps a bank). The digital signature will provide integrity and dissuade repudiation. The 10 digital signature is evidence that the originator did create the event and that the event actually happened. This is useful in case the recipient of the request wants to prove that the originator requested it rather than some impersonator.

150  
155  
160  
165  
170  
175  
180  
185  
190  
195  
200  
205  
210  
215  
220  
225  
230  
235  
240  
245  
250  
255  
260  
265  
270  
275  
280  
285  
290  
295  
300  
305  
310  
315  
320  
325  
330  
335  
340  
345  
350  
355  
360  
365  
370  
375  
380  
385  
390  
395  
400  
405  
410  
415  
420  
425  
430  
435  
440  
445  
450  
455  
460  
465  
470  
475  
480  
485  
490  
495  
500  
505  
510  
515  
520  
525  
530  
535  
540  
545  
550  
555  
560  
565  
570  
575  
580  
585  
590  
595  
600  
605  
610  
615  
620  
625  
630  
635  
640  
645  
650  
655  
660  
665  
670  
675  
680  
685  
690  
695  
700  
705  
710  
715  
720  
725  
730  
735  
740  
745  
750  
755  
760  
765  
770  
775  
780  
785  
790  
795  
800  
805  
810  
815  
820  
825  
830  
835  
840  
845  
850  
855  
860  
865  
870  
875  
880  
885  
890  
895  
900  
905  
910  
915  
920  
925  
930  
935  
940  
945  
950  
955  
960  
965  
970  
975  
980  
985  
990  
995  
1000  
1005  
1010  
1015  
1020  
1025  
1030  
1035  
1040  
1045  
1050  
1055  
1060  
1065  
1070  
1075  
1080  
1085  
1090  
1095  
1100  
1105  
1110  
1115  
1120  
1125  
1130  
1135  
1140  
1145  
1150  
1155  
1160  
1165  
1170  
1175  
1180  
1185  
1190  
1195  
1200  
1205  
1210  
1215  
1220  
1225  
1230  
1235  
1240  
1245  
1250  
1255  
1260  
1265  
1270  
1275  
1280  
1285  
1290  
1295  
1300  
1305  
1310  
1315  
1320  
1325  
1330  
1335  
1340  
1345  
1350  
1355  
1360  
1365  
1370  
1375  
1380  
1385  
1390  
1395  
1400  
1405  
1410  
1415  
1420  
1425  
1430  
1435  
1440  
1445  
1450  
1455  
1460  
1465  
1470  
1475  
1480  
1485  
1490  
1495  
1500  
1505  
1510  
1515  
1520  
1525  
1530  
1535  
1540  
1545  
1550  
1555  
1560  
1565  
1570  
1575  
1580  
1585  
1590  
1595  
1600  
1605  
1610  
1615  
1620  
1625  
1630  
1635  
1640  
1645  
1650  
1655  
1660  
1665  
1670  
1675  
1680  
1685  
1690  
1695  
1700  
1705  
1710  
1715  
1720  
1725  
1730  
1735  
1740  
1745  
1750  
1755  
1760  
1765  
1770  
1775  
1780  
1785  
1790  
1795  
1800  
1805  
1810  
1815  
1820  
1825  
1830  
1835  
1840  
1845  
1850  
1855  
1860  
1865  
1870  
1875  
1880  
1885  
1890  
1895  
1900  
1905  
1910  
1915  
1920  
1925  
1930  
1935  
1940  
1945  
1950  
1955  
1960  
1965  
1970  
1975  
1980  
1985  
1990  
1995  
2000  
2005  
2010  
2015  
2020  
2025  
2030  
2035  
2040  
2045  
2050  
2055  
2060  
2065  
2070  
2075  
2080  
2085  
2090  
2095  
2100  
2105  
2110  
2115  
2120  
2125  
2130  
2135  
2140  
2145  
2150  
2155  
2160  
2165  
2170  
2175  
2180  
2185  
2190  
2195  
2200  
2205  
2210  
2215  
2220  
2225  
2230  
2235  
2240  
2245  
2250  
2255  
2260  
2265  
2270  
2275  
2280  
2285  
2290  
2295  
2300  
2305  
2310  
2315  
2320  
2325  
2330  
2335  
2340  
2345  
2350  
2355  
2360  
2365  
2370  
2375  
2380  
2385  
2390  
2395  
2400  
2405  
2410  
2415  
2420  
2425  
2430  
2435  
2440  
2445  
2450  
2455  
2460  
2465  
2470  
2475  
2480  
2485  
2490  
2495  
2500  
2505  
2510  
2515  
2520  
2525  
2530  
2535  
2540  
2545  
2550  
2555  
2560  
2565  
2570  
2575  
2580  
2585  
2590  
2595  
2600  
2605  
2610  
2615  
2620  
2625  
2630  
2635  
2640  
2645  
2650  
2655  
2660  
2665  
2670  
2675  
2680  
2685  
2690  
2695  
2700  
2705  
2710  
2715  
2720  
2725  
2730  
2735  
2740  
2745  
2750  
2755  
2760  
2765  
2770  
2775  
2780  
2785  
2790  
2795  
2800  
2805  
2810  
2815  
2820  
2825  
2830  
2835  
2840  
2845  
2850  
2855  
2860  
2865  
2870  
2875  
2880  
2885  
2890  
2895  
2900  
2905  
2910  
2915  
2920  
2925  
2930  
2935  
2940  
2945  
2950  
2955  
2960  
2965  
2970  
2975  
2980  
2985  
2990  
2995  
3000  
3005  
3010  
3015  
3020  
3025  
3030  
3035  
3040  
3045  
3050  
3055  
3060  
3065  
3070  
3075  
3080  
3085  
3090  
3095  
3100  
3105  
3110  
3115  
3120  
3125  
3130  
3135  
3140  
3145  
3150  
3155  
3160  
3165  
3170  
3175  
3180  
3185  
3190  
3195  
3200  
3205  
3210  
3215  
3220  
3225  
3230  
3235  
3240  
3245  
3250  
3255  
3260  
3265  
3270  
3275  
3280  
3285  
3290  
3295  
3300  
3305  
3310  
3315  
3320  
3325  
3330  
3335  
3340  
3345  
3350  
3355  
3360  
3365  
3370  
3375  
3380  
3385  
3390  
3395  
3400  
3405  
3410  
3415  
3420  
3425  
3430  
3435  
3440  
3445  
3450  
3455  
3460  
3465  
3470  
3475  
3480  
3485  
3490  
3495  
3500  
3505  
3510  
3515  
3520  
3525  
3530  
3535  
3540  
3545  
3550  
3555  
3560  
3565  
3570  
3575  
3580  
3585  
3590  
3595  
3600  
3605  
3610  
3615  
3620  
3625  
3630  
3635  
3640  
3645  
3650  
3655  
3660  
3665  
3670  
3675  
3680  
3685  
3690  
3695  
3700  
3705  
3710  
3715  
3720  
3725  
3730  
3735  
3740  
3745  
3750  
3755  
3760  
3765  
3770  
3775  
3780  
3785  
3790  
3795  
3800  
3805  
3810  
3815  
3820  
3825  
3830  
3835  
3840  
3845  
3850  
3855  
3860  
3865  
3870  
3875  
3880  
3885  
3890  
3895  
3900  
3905  
3910  
3915  
3920  
3925  
3930  
3935  
3940  
3945  
3950  
3955  
3960  
3965  
3970  
3975  
3980  
3985  
3990  
3995  
4000  
4005  
4010  
4015  
4020  
4025  
4030  
4035  
4040  
4045  
4050  
4055  
4060  
4065  
4070  
4075  
4080  
4085  
4090  
4095  
4100  
4105  
4110  
4115  
4120  
4125  
4130  
4135  
4140  
4145  
4150  
4155  
4160  
4165  
4170  
4175  
4180  
4185  
4190  
4195  
4200  
4205  
4210  
4215  
4220  
4225  
4230  
4235  
4240  
4245  
4250  
4255  
4260  
4265  
4270  
4275  
4280  
4285  
4290  
4295  
4300  
4305  
4310  
4315  
4320  
4325  
4330  
4335  
4340  
4345  
4350  
4355  
4360  
4365  
4370  
4375  
4380  
4385  
4390  
4395  
4400  
4405  
4410  
4415  
4420  
4425  
4430  
4435  
4440  
4445  
4450  
4455  
4460  
4465  
4470  
4475  
4480  
4485  
4490  
4495  
4500  
4505  
4510  
4515  
4520  
4525  
4530  
4535  
4540  
4545  
4550  
4555  
4560  
4565  
4570  
4575  
4580  
4585  
4590  
4595  
4600  
4605  
4610  
4615  
4620  
4625  
4630  
4635  
4640  
4645  
4650  
4655  
4660  
4665  
4670  
4675  
4680  
4685  
4690  
4695  
4700  
4705  
4710  
4715  
4720  
4725  
4730  
4735  
4740  
4745  
4750  
4755  
4760  
4765  
4770  
4775  
4780  
4785  
4790  
4795  
4800  
4805  
4810  
4815  
4820  
4825  
4830  
4835  
4840  
4845  
4850  
4855  
4860  
4865  
4870  
4875  
4880  
4885  
4890  
4895  
4900  
4905  
4910  
4915  
4920  
4925  
4930  
4935  
4940  
4945  
4950  
4955  
4960  
4965  
4970  
4975  
4980  
4985  
4990  
4995  
5000  
5005  
5010  
5015  
5020  
5025  
5030  
5035  
5040  
5045  
5050  
5055  
5060  
5065  
5070  
5075  
5080  
5085  
5090  
5095  
5100  
5105  
5110  
5115  
5120  
5125  
5130  
5135  
5140  
5145  
5150  
5155  
5160  
5165  
5170  
5175  
5180  
5185  
5190  
5195  
5200  
5205  
5210  
5215  
5220  
5225  
5230  
5235  
5240  
5245  
5250  
5255  
5260  
5265  
5270  
5275  
5280  
5285  
5290  
5295  
5300  
5305  
5310  
5315  
5320  
5325  
5330  
5335  
5340  
5345  
5350  
5355  
5360  
5365  
5370  
5375  
5380  
5385  
5390  
5395  
5400  
5405  
5410  
5415  
5420  
5425  
5430  
5435  
5440  
5445  
5450  
5455  
5460  
5465  
5470  
5475  
5480  
5485  
5490  
5495  
5500  
5505  
5510  
5515  
5520  
5525  
5530  
5535  
5540  
5545  
5550  
5555  
5560  
5565  
5570  
5575  
5580  
5585  
5590  
5595  
5600  
5605  
5610  
5615  
5620  
5625  
5630  
5635  
5640  
5645  
5650  
5655  
5660  
5665  
5670  
5675  
5680  
5685  
5690  
5695  
5700  
5705  
5710  
5715  
5720  
5725  
5730  
5735  
5740  
5745  
5750  
5755  
5760  
5765  
5770  
5775  
5780  
5785  
5790  
5795  
5800  
5805  
5810  
5815  
5820  
5825  
5830  
5835  
5840  
5845  
5850  
5855  
5860  
5865  
5870  
5875  
5880  
5885  
5890  
5895  
5900  
5905  
5910  
5915  
5920  
5925  
5930  
5935  
5940  
5945  
5950  
5955  
5960  
5965  
5970  
5975  
5980  
5985  
5990  
5995  
6000  
6005  
6010  
6015  
6020  
6025  
6030  
6035  
6040  
6045  
6050  
6055  
6060  
6065  
6070  
6075  
6080  
6085  
6090  
6095  
6100  
6105  
6110  
6115  
6120  
6125  
6130  
6135  
6140  
6145  
6150  
6155  
6160  
6165  
6170  
6175  
6180  
6185  
6190  
6195  
6200  
6205  
6210  
6215  
6220  
6225  
6230  
6235  
6240  
6245  
6250  
6255  
6260  
6265  
6270  
6275  
6280  
6285  
6290  
6295  
6300  
6305  
6310  
6315  
6320  
6325  
6330  
6335  
6340  
6345  
6350  
6355  
6360  
6365  
6370  
6375  
6380  
6385  
6390  
6395  
6400  
6405  
6410  
6415  
6420  
6425  
6430  
6435  
6440  
6445  
6450  
6455  
6460  
6465  
6470  
6475  
6480  
6485  
6490  
6495  
6500  
6505  
6510  
6515  
6520  
6525  
6530  
6535  
6540  
6545  
6550  
6555  
6560  
6565  
6570  
6575  
6580  
6585  
6590  
6595  
6600  
6605  
6610  
6615  
6620  
6625  
6630  
6635  
6640  
6645  
6650  
6655  
6660  
6665  
6670  
6675  
6680  
6685  
6690  
6695  
6700  
6705  
6710  
6715  
6720  
6725  
6730  
6735  
6740  
6745  
6750  
6755  
6760  
6765  
6770  
6775  
6780  
6785  
6790  
6795  
6800  
6805  
6810  
6815  
6820  
6825  
6830  
6835  
6840  
6845  
6850  
6855  
6860  
6865  
6870  
6875  
6880  
6885  
6890  
6895  
6900  
6905  
6910  
6915  
6920  
6925  
6930  
6935  
6940  
6945  
6950  
6955  
6960  
6965  
6970  
6975  
6980  
6985  
6990  
6995  
7000  
7005  
7010  
7015  
7020  
7025  
7030  
7035  
7040  
7045  
7050  
7055  
7060  
7065  
7070  
7075  
7080  
7085  
7090  
7095  
7100  
7105  
7110  
7115  
7120  
7125  
7130  
7135  
7140  
7145  
7150  
7155  
7160  
7165  
7170  
7175  
7180  
7185  
7190  
7195  
7200  
7205  
7210  
7215  
7220  
7225  
7230  
7235  
7240  
7245  
7250  
7255  
7260  
7265  
7270  
7275  
7280  
7285  
7290  
7295  
7300  
7305  
7310  
7315  
7320  
7325  
7330  
7335  
7340  
7345  
7350  
7355  
7360  
7365  
7370  
7375  
7380  
7385  
7390  
7395  
7400  
7405  
7410  
7415  
7420  
7425  
7430  
7435  
7440  
7445  
7450  
7455  
7460  
7465  
7470  
7475  
7480  
7485  
7490  
7495  
7500  
7505  
7510  
7515  
7520  
7525  
7530  
7535  
7540  
7545  
7550  
7555  
7560  
7565  
7570  
7575  
7580  
7585  
7590  
7595  
7600  
7605  
7610  
7615  
7620  
7625  
7630  
7635  
7640  
7645  
7650  
7655  
7660  
7665  
7670  
7675  
7680  
7685  
7690  
7695  
7700  
7705  
7710  
7715  
7720  
7725  
7730  
7735  
7740  
7745  
7750  
7755  
7760  
7765  
7770  
7775  
7780  
7785  
7790  
7795  
7800  
7805  
7810  
7815  
7820  
7825  
7830  
7835  
7840  
7845  
7850  
7855  
7860  
7865  
7870  
7875  
7880  
7885  
7890  
7895  
7900  
7905  
7910  
7915  
7920  
7925  
7930  
7935  
7940  
7945  
7950  
7955  
7960  
7965  
7970  
7975  
7980  
7985  
7990  
7995  
8000  
8005  
8010  
8015  
8020  
8025  
8030  
8035  
8040  
8045  
8050  
8055  
8060  
8065  
8070  
8075  
8080  
8085  
8090  
8095  
8100  
8105  
8110  
8115  
8120  
8125  
8130  
8135  
8140  
8145  
8150  
8155  
8160  
8165  
8170  
8175  
8180  
8185  
8190  
8195  
8200  
8205  
8210  
8215  
8220  
8225  
8230  
8235  
8240  
8245  
8250  
8255  
8260  
8265  
8270  
8275  
8280  
8285  
8290  
8295  
8300  
8305  
8310  
8315  
8320  
8325  
8330  
8335  
8340  
8345  
8350  
8355  
8360  
8365  
8370  
8375  
8380  
8385  
8390  
8395  
8400  
8405  
8410  
8415  
8420  
8425  
8430  
8435  
8440  
8445  
8450  
8455  
8460  
8465  
8470  
8475  
8480  
8485  
8490  
8495  
8500  
8505  
8510  
8515  
8520  
8525  
8530  
8535  
8540  
8545  
8550  
8555  
8560  
8565  
8570  
8575  
8580  
8585  
8590  
8595  
8600  
8605  
8610  
8615  
8620  
8625  
8630  
8635  
8640  
8645  
8650  
8655  
8660  
8665  
8670  
8675  
8680  
8685  
8690  
8695  
8700  
8705  
8710  
8715  
8720  
8725  
8730  
8735  
8740  
8745  
8750  
8755  
8760  
8765  
8770  
8775  
8780  
8785  
8790  
8795  
8800  
8805  
8810  
8815  
8820  
8825  
8830  
8835  
8840  
8845  
8850  
8855  
8860  
8865  
8870  
8875  
8880  
8885  
8890  
8895  
8900  
8905  
8910  
8915  
8920  
8925  
8930  
8935  
8940  
8945  
8950  
8955  
8960  
8965  
8970  
8975  
8980  
8985  
8990  
8995  
9000  
9005  
9010  
9015  
9020  
9025  
9030  
9035  
9040  
9045  
9050  
9055  
9060  
9065  
9070  
9075  
9080  
9085  
9090  
9095  
9100  
9105  
9110  
9115  
9120  
9125  
9130  
9135  
9140  
9145  
9150  
9155  
9160  
9165  
9170  
9175  
9180  
9185  
9190  
9195  
9200  
9205  
9210  
9215  
9220  
9225  
9230  
9235  
9240  
9245  
9250  
9255  
9260  
9265  
9270  
9275  
9280  
9285  
9290  
9295  
9300  
9305  
9310  
9315  
9320  
9325  
9330  
9335  
9340  
9345  
9350  
9355  
9360  
9365  
9370  
9375  
9380  
9385  
9390  
9395  
9400  
9405  
9410  
9415  
9420  
9425  
9430  
9435  
9440  
9445  
9450  
9455  
9460  
9465  
9470  
9475  
9480  
948

Additionally, the need to protect keys against exposure can be different. In a workstation environment, exposure of keys could cause problems with the originator's resources, but the damage would be confined to this one person, and the cost of that damage contained. On the other hand, in a server, compromise of the keys could jeopardize all the users and their accounts.

5 For example, one is generally willing to spend more effort protecting a bank's resources than one is willing to spend to protect an individual's resources.

In the past, the problem of the server environment has been addressed by a variety of efforts. Application designers are faced with a dilemma of how to protect these secrets (typically cryptographic keys). Storing them in a text file, or within a program, or even in an independent 10 box, is an open invitation to fraud. Text files can be copied and examined easily by most anyone with access to the computer (in fact, really protecting a file system is difficult). Keys stored in programs can make the program files themselves a valuable target of fraud. Keys stored in independent boxes become vulnerable because it is difficult to control which programs or which agents access the box.

Furthermore, relying upon firewalls helps protect the secrets against external attack, but leaves unguarded fraud from corruptible employees. Traditional banking practice requires multiple individuals to perform certain tasks, such as opening a vault. To enforce this requirement, separate keys are entrusted to separate trusted officers, while requiring both keys to unlock the vault.

20 The prior art public key infrastructure (PKI) schemes are built upon a "workstation paradigm." The workstation paradigm has an individual user, at a given workstation, that utilizes encryption technology on the workstation to send encrypted messages to another person, or themselves at another workstation. In the workstation paradigm, servers are used only as transport mechanisms. The advent of the worldwide web of the Internet has eroded many of the 25 underlying assumptions of the workstation paradigm. Unfortunately, the prior art PKI schemes have not kept pace. There is, therefore, a need for a crypto-system that allows multiple simultaneous users having multiple sessions while preserving security and integrity of both keys, signatures, access rights, and an apparatus and method to enable automated trust relationships on computer server for multiple applications and multiple users.

## SUMMARY OF THE INVENTION

The present invention provides a paradigm shift from the workstation-centric cryptographic system to a server-centric cryptographic system.

The present invention utilizes a Key Repository process that is designed to link at least 5 two cryptographic keys in order to dissuade tampering and to detect any such tampering with sensitive information that is kept on a database. The Key Repository of the present invention, therefore, addresses the management of trust within an organization. The Key Repository process of the present invention does not replace the traditional functions performed by firewalls and sound security policy, it simply augments them. A Key Repository process is initiated with 10 human action, and after the necessary steps of authentication and authorization, the keying material is made available to this process. The Key Repository process is the only program in the computer system that knows the critical secrets. The Key Repository process will supply selected keying material to pre-authorized applications, thus limiting the spread of the secret information, and eliminating the need for human intervention after system startup. The Key 15 Repository process enforces policy decisions in such areas as identifying authorized applications, changes in parameters, and does so by requiring multiple approvals before changes are implemented. In addition, software programs can be pre-authenticated to act as an extension to the Key Repository process.

The Key Repository process never records sensitive data in the clear on disks, avoiding 20 the problems that could occur if there was any unauthorized access to the disk storage, or to the disk or backup media. Should someone have access to such disk or backup media, uncovering the secrets would be infeasible since the attacker would have to have access to multiple distributed passwords. The memory and internal data transfer paths are presumed to be secure 25 enough to handle the movement of sensitive data. The Key Repository process also enforces a variety of security policies, such as authenticating operators and owners, controlling the lifetime and quality of generated keys, requiring multiple approvals before changing security parameters, etc.

Application programs, implementing the customer's business functions, may request 30 copies of keys or other cryptographic secrets. To prevent some rogue or unauthorized programs from obtaining keys, each program entitled to receive keys must be authorized by the key owners. This removes the necessity for programmers to embed keys within a program, to store

them in the clear on data files, or to be present to supply passwords when the application program is initiated.

In most environments, the Key Repository process can be expected to be present and provide the needed services on demand, allowing it to partake in mission-critical business 5 applications. The Key Repository process can thus be expected to survive even in the face of the usual single points of failure that are allowed for by the computer system. This means that the human restart of the Key Repository process is needed only rarely.

Using commands or directives issued to Key Repository process, the customer's staff interacts with the Key Repository process. Many of the activities are considered *security related*, 10 in that they involve the extension of trust. For these activities to become effective, the approval of one or more individuals is required. Initially, the system requires just one approved individual, but this number can be changed to any number (the change itself is of course a security-related operation). For example, the customer might want to require that any five out of nine key owners to approve of certain security parameters. These key owners are individuals 15 with a name and password, known to the Key Repository process' database.

Security related operations include the following types of functions:

- a) Changing the number of approvals required;
- b) Adding or deleting operators or owners;
- c) Pre-authenticating an application program as being allowed to obtain certain 20 cryptographic secrets;
- d) Modifying the cryptographic algorithms in use; and
- e) Pre-authenticating a program to act as an extension of the Key Repository process.

These features permit the enterprise to manage and control e-commerce applications, while enforcing the security policy that fits the business model. Authorized application 25 programs have access to the cryptographic secrets needed to fulfill the business functions, but no single individual (or small group of individuals) can easily compromise the security of the system.

Hence, in accordance with the purpose of the present invention as embodied and broadly described herein, the present invention relates to a system and method for linking multiple 30 cryptographic keys. Features of the invention include at least one process; two or more master

keys; and a key repository process. Of the two or more master keys one master key is more secure than the other master keys and requires a multi-part construction to be exposed, wherein, to be exposed, the other master keys require construction from fewer parts than the more secure master key. The key repository process maintains and pre-authorizes access to the two or more master keys by any selected process from among the at least one process. The key repository process, provides for linking the more secure master key with one or more of the other master keys in order to detect any key tampering.

The key repository process performs the linking by creating a message digest of the more secure master key concatenated with a less secure one of the other master keys, the linking result being saved in a database. Alternatively, the key repository process performs the linking by creating a message digest of the more secure master key concatenated with a random value and further concatenated with a less secure one of the other master keys, the linking result being saved in a database. Preferably, the random value is a Salt. In yet another alternative, the key repository process performs the linking by using the more secure master key as a symmetric encryption key, to compute a symmetric authentication code, and retaining some or all of the linking result. In yet another alternative the key repository process performs the linking to produce an 8-byte linking result by using the more secure master key as a symmetric encryption key, to compute a symmetric authentication code, and retaining a 4-byte portion of the linking result.

20

#### BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of the processes of the present invention;

Figure 2 is a flow chart of the Key Repository process access method of the present invention;

25 Figure 3 is a flow chart of the initial key creation method of the present invention;

Figure 4 is a block diagram of an alternate embodiment of the present invention;

Figure 5 is a block diagram of an alternate embodiment of the present invention; and

Figure 6 is a flow chart of the method of establishing a cryptographic link between the master keys of the present invention.

#### DETAILED DESCRIPTION OF THE INVENTION

5 The present invention solves the problems inherent in the prior art by providing an apparatus and method for mimicking the human login process for specific and individual processes on a server-based computer system. Although the present invention may be deployed on a wide variety of computer architectures, in its best mode, the present invention is used on high-availability servers, such as the Non-Stop® Himalaya server systems produced by the  
10 Compaq Computer Corporation of Houston, Texas.

The method and apparatus of the present invention enable end entities to conduct commerce over unprotected networks (such as the Internet) in a secure fashion. In cryptographic parlance, an end entity is a person, router, server, software process, or other entity that uses a digital certificate to identify itself. In the context of the present invention, the definition of "consumer" includes any end entity. However, the definition of "consumer" for the present invention is broader in that it can include both people and organizations (both *de facto* and *de jure*). "Consumer" is, interchangeably, also referred to herein as a "client." In this sense, the consumer of the present invention can be an individual (with or without their own digital certificate) sitting at home trying to access their bank accounts. It can also be a business enterprise that has an automated software agent purchasing products based upon some pre-defined criteria. It will be apparent to those skilled in the art that the consumer of the present invention can be a wide variety of entities (real, legal, and abstract).

25 The term "enterprise," in the context of the present invention, has essentially the same definition of "consumer." A separate term is used in order to illustrate more clearly the two sides of a transaction using the present invention. In practice, however, a consumer is typically a person or small business organization. An enterprise can be a business organization of any size. While this is not necessarily always the case, for purposes of the following illustration, a consumer can be thought of as an individual person and an enterprise can be thought of as a business organization even if owned/operated by one person.

30

#### The Processes of the Present Invention

The basic architecture of the present invention includes several components as shown in Figure 1. The computer system 10 can be composed of one or more computer servers or other computing devices. The only requirement of the server or device is that it be able to run one or more of the processes of the present invention. If only one server is used, it must be capable of 5 executing all of the processes of the present invention. Otherwise, if multiple servers are used, then the processes may be "farmed out" or instantiated on separate servers that are connected to a common network. In an embodiment of the present invention, it is assumed that the transmissions between the servers of the computer system 10 can be made in a secure manner. The common network mentioned above is not intended to be something like the Internet, unless 10 extra transmission-related security measures (such as SSL (Secure Socket Layer), SSH (Secure Shell) or IPSec (Internet Protocol Security), etc.) are invoked.

The present invention uses cryptographic keys to manage confidential information on a database. Two keys form a set of master keys in the present invention. Each of the two master keys performs a specific function within the cryptographic scheme of the present invention. One of the keys is configured within a Key Repository process to maintain the integrity of critical information on the database. The other master key is used within the Key Repository process to protect the confidential information on the database.

An alternate embodiment of the present invention can utilize one master key in the Key Repository process. This single master key would be used both for integrity of the critical information on the database as well as to protect the confidential information. However, use of a 20 single key may give the user pause to let operators have access to a complete key. Therefore, a single master key may require reformulation of the administrative procedures for the generation and use of the single master key.

According to Figure 1, the Key Repository process 20 is a process that runs on the 25 computer system 10. The Key Repository process 20 makes use of a database 30 to save necessary information. Included on this database 30 are one or more entries defining Operators, and two or more entries defining Owners. The Operator entries are used to retain the value of the Integrity Key 22. The Integrity Key 22 is configured to ensure the integrity of sensitive information within the database 30. The Owner entries are used to retain a share (described 30 below) of the Protection Key 24. The Protection Key 24 is configured to protect sensitive information on the database 30. In both cases, the entries protect their respective secret by a

password-based public-key encryption method described below. Finally, database 30 also stores the enterprise credentials 32 of the enterprise 31 that sponsors the application 40. In this illustration, the consumer 50 uses the applications 40 to conduct business with the enterprise 31 whose enterprise credentials 32 are retrieved from the database 30, thus allowing applications 40 (representing the enterprise 31) and consumer 50 to conduct business. This business transaction can be two-way in that the application 40 can be the business conduit on behalf of either or both the consumer 50 and the enterprise 31.

#### Password-Based Public-Key Encryption

When the Operator or Owner is introduced to the system, the process of the present invention first obtains a name and a password from the command interface. Second, the process invents a salt value and a public/private RSA key pair. None of these values are revealed to the individual. The RSA encryption scheme requires the use of two keys, one of which is usually public and is not hidden, the other that is private. For purposes of the present invention, a salt value is a random number whose value is known and saved, but itself has no information encoded in it. A salt value is also sometimes called a nonce.

Third, the Operator's or Owner's password is combined with the salt value to compute a symmetric key and initialization vector, and, using any of a number of symmetric encryption algorithms in cipher block chaining (CBC) mode, the operator's or owner's RSA private key that was invented above, is encrypted. The methodology for this is described by RSA's PKCS #5 (password-based encryption), although the method described by RSA has been extended by allowing the customer to specify which hashing algorithm is used, the hash-iteration count, and the symmetric encryption algorithm. For purposes of this application, a hash (also known as a message digest) is a function that takes input data and produces a short answer (typically 128 or 160 bits in length) with the property that, given the answer, it is unfeasible to construct input data, other than the true data, which will produce the same answer. A hash function is often called a "one-way" function. By repeating a hash function, one increases the amount of time it would take to break the one way function by trial-and-error. Thus, hashes are often repeated thousands of times. In addition, the present invention uses the value of the protection key (one of the master keys) to encrypt the RSA public key. This prevents an undetected modification of the protected secret unless the true RSA public key can be exposed.

The secret, which for Operators is the value of the Integrity Key, and for Owner is their share of the value of the Protection Key, is then encrypted by the RSA public key, using the methodology described by RSA's PKCS#1. The database record for the Owner (or Operator) is then written, and contains:

- 5      a) the name of the Operator or Owner;
- b) the salt value;
- c) the public key of the RSA key pair, encrypted with the protection key;
- d) the name of the hashing algorithm used;
- e) the number of times the hash value was re-hashed;
- 10     f) the name of the symmetric algorithm used;
- g) the private key of the RSA key pair (encrypted with password-based encryption); and
- h) the secret value encrypted by the RSA key pair.

As implied above, the Protection Key 24 is divided into shares. In an embodiment, the Bloom-Shamir sharing algorithm is used for the division. The number of shares is dictated by the count of Owners known to the system. The order of the algorithm (i.e., the number of shares that will be needed to reconstitute the secret) is dictated by the current value of the Approval Count parameter, which can be any number between one and one less than the number of Operators known to the computer system 10.

Although the system of the present invention initially is activated with an Approval Count of one, customers are advised to change the approval count to a higher number in accordance with their own security policy. For example, a banking institution could require that seven out of fifty individuals be required to recreate the bank's Protection Key 24. The use of seven reduces the fraud-risk, and the availability of fifty candidates increases the chances of locating at least seven. In the present invention, the individuals needed to expose the Integrity 25 Key 22 need not be the same set of individuals needed to re-compute the Protection Key 24.

The crucial information in the database 30 is protected against modification by an Integrity Key 22. The confidential data is protected by the Protection Key 24. When the Key Repository process 20 is restarted, one of the Operators known to the system exposes the Integrity Key 22 by use of the correct identity and password, as described below. The Protection 30 Key 24 is assembled from a set of secrets that are split among multiple individuals, known as Owners, according to the Bloom-Shamir methodology. When the requisite number of Owners

have exposed their share of the split Protection Key 24, using the methodology described below, the Protection Key 24 can be recovered.

#### Operators and Owners

5 In the present invention, two distinct classes of system operators are needed. Operators are responsible for the overall computing environment. Any operator can start the Key Repository process, and in so doing, asserts that the computer system is indeed what it claims to be. In alternate embodiments of this invention, this assertion is used to unlock and expose a set of cryptographic credentials with which the Key Repository process can communicate in a  
10 trusted and secure fashion with resources external to the platform, allowing that resource to be, but not limited to, the following set of functions:

- a) A remotely located owner can perform his functions using a secure link; and
- b) A remote extension of the Key Repository process (Remote Agent), or a remote Key Repository process can act on behalf of the local Key Repository process and perform those actions which require access local to the application program.

#### Initial Key Creation

When the Key Repository process 20 is executed for the first time, that is, without a database 30 from a prior run, the person initiating the process provides the program with the following information:

1. the name of the database 30;
2. the name and password for one operator;
3. the name and password for one owner; and
4. the name and password for a second owner.

25 The Key Repository process 20 defines these individuals in the database 30, as described above, and creates both a Protection Key 24 and an Integrity Key 22, encoding these values in the database 30 as described above. The values of both the Protection Key 24 and the Integrity Key 22 are then erased from the computer memory, and can be reconstituted by an Operator in conjunction with one or more of the Owners by supplying their respective identity and password.

30 The method of creating the keys is illustrated in Figure 3. The method starts with step 300. Thereafter, the name of the database is provided in step 302. The name and password for at

least one operator is provided in step 304. The names and passwords of at least two owners are provided in steps 306 and 308. Next, in step 310, the Key Repository process creates an Integrity Key. Thereafter, the Key Repository process creates a Protection Key in step 312. It should be noted that the order of the steps previously discussed is not important except that all of 5 the input information must be provided before all of the keys can be created. For example, after the names and passwords have been entered, either one of the keys can be created. After the keys have been created and recorded on the database using the password-based public-key encryption method defined above, the keys are erased from memory. At this point, the Key Repository process is activated by the Operator, and either (at least one) of the Owners who 10 supply their identity (e.g., their user-name) and their passwords, step 314. The process then ends in step 316.

At this point, because the initial passwords may not have been well protected, it is recommended practice, and part of an embodiment of the present invention, to require that each operator and owner modify his/her respective password without allowing other individuals to see the keystrokes, and that additional Operators and Owners be added to the list of legitimate operators/owners, as required by the customer's security policy, and that other security parameters (such as the approval count, the algorithm selections) are adjusted, as required by the customer's security policy. All of the above suggestions are, however, optional, and it is within the scope of alternate embodiments of the present invention to not use the elements of the above-recommended practice.

In an embodiment of the present invention, the value of the Integrity Key 22, as well as each value of a share of the Protection Key 24, is stored on the database 30 encrypted using the password-based public-key technique. The Operator or Owner by entering his name allows the Key Repository process 20 to find the relevant record in the database 30. The password supplied 25 by the Operator or Owner is combined with a *salt* value that is stored in the database 30. These values allow the Key Repository process 20 to compute the symmetric encryption key,  $E_{\text{symkey}}$ , and an initialization vector IV, as defined by the well known RSA PKCS #5 methodology. These values are then used to decrypt the secret using the methodology defined by RSA PKCS #1. For Operators, the PKCS #1 methodology will expose the Integrity Key 22. For owners, the 30 PKCS #1 methodology will expose their portion of the Protection Key 24. The RSA key pair

alluded to is generated by the Key Repository process 20 at the time that the operator or owner entry is initially defined.

The double encryption password-based public-key encryption technique enables the Key Repository process 20 to change the value of either or both the Integrity Key 22 and the Protection Key 24 once the existing key values are known. Moreover, the changes to the keys can be done without access to operator or owner passwords. In an embodiment of the present invention, this key change occurs in several situations, including:

- 5 a) whenever an operator or owner is added or deleted;
- b) whenever the definition of the algorithms used in the above process is changed; or
- 10 c) whenever the database must be rewritten in order to accommodate an expansion in the size of the database.

Other reasons for changing the key values are institutional regulation (such as government security regulations over banks) and national security issues, which mandate periodic changes in key values.

In addition to the secrets and other information mentioned above, the database 30 also contains policy values and entity credentials that include the enterprise's 31 credentials 32. The database 30 also includes the certificates, private keys and trust root. Trust roots (also known as a trust points) are verification certificates known to be authentic, and which can be used to derive the trustworthiness of other certificates using the methodology defined by the Public Key and Attribute Certificate Frameworks ITU X.509, CCIT: Recommendation X.509, ISO/IEC 9594 which is available from the American National Standards Institute (ANSI) 1819 L Street, NW, Washington, D.C. 20036. Note, the end entity is normally an organization (such as an institution or enterprise). However, alternate embodiments of the present invention include humans in the list of enterprises 32, such as humans or other groupings or entities that own cryptographic keys 25 to be entrusted in a computer program.

The enterprise end-entity 31, whose secrets the Key Repository process 20 is protecting in the form of enterprise credentials 32, is often an institution or enterprise. These secrets are frequently of significant value, and must not be given to unauthorized application programs. For this reason, the Key Repository process 20 records in the database 30 those applications 40 that 30 are authorized to have access to the enterprise credentials 32.

When an application is initially authorized by the key owners (those with portions of the Protection Key 24), the Key Repository process 20 calculates a checksum (akin to the checksum used with Microsoft's Authenticode) that enables the Key Repository process 20 to detect any future modification in the content of the program binary (executable) file. In the embodiment of 5 the present invention, additional commands are available to the owners that direct the Key Repository process 20 to record in the database 30 authorizations that list the enterprise's 31 name and the name of the application 40 and associated program authentication information for which that enterprise's credentials 32 are authorized. This list is used by the Key Repository process 20 to ensure that the application program 40 is authorized to run and to obtain the 10 enterprise credentials 32 of the specific enterprise 31.

Unlike prior art cryptographic schemes where all of the vital keys, or the passwords to unlock those keys, are kept in memory, and where that memory could be recorded on some storage device (perhaps by operating-system functions such as swapping), in an embodiment of the present invention, the clear version of these vital keys and passwords are retained in protected physical memory. In this way, the security of the system is enhanced by making it impractical to recover the keys from non-volatile storage. However, alternate embodiments of the present invention can keep the keys in virtual memory.

#### Processing Requests for Enterprise End-Entity Keys

Alternate embodiments of the present invention are illustrated in Figures 4 and 5. The 20 architecture of the computer system 10 of Figure 4 is similar to that illustrated in Figure 1. However, this alternate embodiment adds an extra interface 26 to the Key Repository process 20. The local interface 26 is used by applications 40 to communicate with the Key Repository process 20 where all of the processes reside on the same computer system 10 as illustrated in 25 Figure 4.

The alternate embodiment that is illustrated in Figure 5, however, differs from that in Figure 4 in that some of the applications 40 are instantiated on a second computer system 11. As a result, a different authorization/authentication mechanism is required to process the request for enterprise credentials 32. The applications 40 interact with the local interface 26 on the local 30 agent 21 which can, in yet another alternate embodiment, act as a local Key Repository process. The local agent 21, then, via a secure mechanism (such as SSH, SSL, IP tunneling, etc.) interacts

with the Key Repository process 20 via the remote interface 28 as shown in Figure 5. In this way, the Key Repository process 20 can interact with multiple remote computer systems and thereby extend the scope of capability of the present invention wherein the scalability of the present invention is limited only by the hardware and bandwidth resources.

5 It should be noted, however, that remote computer systems, such as the computer system 11 of Figure 5, must have been pre-authenticated and have established a secure connection. Typically, this involves having some human beings (typically Operators) at both ends of the connection to authenticate the machines (i.e., computer systems 10 and 11).

10 Referring to Figure 5, a typical business transaction starts with a consumer 50 in the form of, for example, some person making use of a workstation to formulate a request. Included in this request would be the information necessary to authenticate the consumer (such as a user-name and password and/or a digital certificate), and to transmit that authentication information to the server on the computer system 11. A connection is made to the business application 40, which resides in a server on the computer system 11 that is disjoint from the Key Repository process 20 (that resides on the computer system 10). The application 40, determining that it needs the enterprise credentials (keys) 32 of the enterprise entity 31 that the application 40 is representing, sends such a request to the Key Repository process' 20 local agent 21. From the perspective of the application 40, the use of the local interface 26 on the local agent 21 is identical to the local interface 26 of the Key Repository process 20. Thus the local agent 21 can correctly be characterized as a remote Key Repository process. Thus an application 40 of Figure 20 4 would behave identically to an application 40 of Figure 5. The request from the application 40 is received by the local agent 21. The local agent 21 communicates, using a secure communications medium, with the actual Key Repository process 20, which resides within the computer system 10. In this communication, the local agent 21 includes the identification and 25 authentication information of the application process 40, whereby the Key Repository process 20 will be able to determine with assurance which application is making the request. The communication terminates at the local agent remote agent interface 28 of the Key Repository process 20 as illustrated in Figure 5.

30 In both cases illustrated in Figures 4 and 5, the Key Repository process 20 now knows with high certainty: 1) the identity of the application program 40 making the request; and 2) the identity of the enterprise 31 whose secrets (enterprise credentials (keys) 32) are being requested.

The decision whether to grant or to refuse the request is made based upon the information stored in the Key Repository process' database 30. In this database 30, the integrity of information is protected by the Integrity Key 22, and confidential information (such as private keys) is protected by the Protection Key 24. Both of these values are known to the Key

5 Repository process 20, and thus the decision can be then rendered based upon authenticated data.

If the decision is made to grant the access, the enterprise credentials 32 are extracted from the database 30, decrypted using the Protection Key 24, re-encrypted, and returned to the requesting application 40 as a response along with a password to decrypt the data. In the example illustrated in Figure 5, the response will be protected during the transit to the originating 10 system 11 using the secure connection between them.

### Operation of the Present Invention

#### Control of Enterprise Credentials and Authentication of Application Programs

The method of the present invention is illustrated in the flow chart of Figure 2. This particular method of the present invention presumes that a Key Repository process has been instantiated and that the needed values are already present in the database. Referring to Figure 2, the process begins at step 200. An application server is instantiated on the computer system (having the Key Repository process and database) in step 202. The application server is a process within the computer system which executes all or part of the desired application. There may be one or more instances of this application server process. This process may have been activated by the arrival of some work item from the outside (not shown), or may have been pre-initiated and be waiting for work.

Next, in step 204, work arrives that is to be performed by the application on behalf of one of the consumers known to the system. The application contacts the Key Repository process in 25 step 206. Next, in step 208, the Key Repository process attempts to authenticate the application, i.e., to determine if the application has been tampered with since it was first registered with the Key Repository process. If the application has not been tampered with (and is one of the applications that is allowed access to the Key Repository process) it is granted access to the Key Repository process, and execution continues to step 210. Otherwise, access to the Key Repository process is denied and execution skips to the end with, optionally, an appropriate error 30 message in step 213 and termination in step 220. If the application has been authenticated, it

next requests the enterprise credentials of the enterprise on whose behalf the application is tasked to act, step 210. The Key Repository process then checks, in step 212, whether the requesting application is allowed to have access to the enterprise's credentials. If the Key Repository process determines that access is to be denied, an indication of that result is returned to the requesting application, which, after optionally indicating an error in step 213 and terminates at step 220. If, however, access is to be approved, execution continues to step 214, where the Key Repository process builds a credential file of the end entity for use by the application. Thereafter, in step 216, the Key Repository process transmits (gives) the new credential file, to the application along with a password that enables the application to perform its mission. Thus 10 enabled, the application performs the work on behalf of the consumer (and/or the enterprise) in step 218. In any case, the execution of this method ends in step 220.

#### Enterprise End-Entity Key Rollover

The Key Repository process is entrusted with the safe-keeping of the private keys and trust roots of an enterprise end-entity. The public keys are contained in digital certificates signed by an issuing authority (such as a Certification Authority). There are several situations which can occur that warrant the potential re-issuance of the certificates. These situations include but are not limited to:

- a) the approach of the expiration date of the certificate, much as a driver's license is renewed prior to its expiration date;
- b) a request by the issuing authority to re-issue the certificate, implying that the certificate owner must periodically check some external database; and
- c) a notification from the issuing authority to renew the certificate.

Any of these conditions can cause the Key Repository process, which owns the enterprise end-entity's certificates, to initiate a set of certificate management functions.

In the workstation model, this is usually performed when the user goes through manual logon, and when the individual's keys are currently not in use. In the server model, however, the keys can be in use at any time, and a "logon" event never really happens.

30 In an embodiment of the current invention, this issue is addressed directly by the Key Repository process. On a time periodic basis, typically once per day but adjustable by the customer, the Key Repository process will check to see if a certificate re-issuance is called for.

While this check is going on, and until it is resolved, one way or another, the old keys remain available to applications. During this check, the Key Repository process may utilize the services of a specialized application that acts as an extension of the Key Repository process, but if so, that application has been duly authenticated and authorized to perform such a function.

5 If the rollover operation results in new or revised keys, the Key Repository process will store these new values in its database. When that storage operation is completed, subsequent requests from applications will receive the new key values. While the applications are still running, however, they can continue to use the older keys. Such requests from application programs can be stimulated by notification, timeout, encountering stale keys, or other events.

10

#### Secure Preservation of Intermediate State

In addition to the above activities, the Key Repository process 20 (of Figure 1) maintains a set of symmetric keys that are supplied specifically to authenticated application programs 40. These symmetric keys allow the application to save intermediate sensitive data to an application-defined database (not shown), or entrust them to insecure cookies (as described below), such that subsequent activities on behalf of the same piece of work can be performed by other application programs 40. This permits the applications 40 to be implemented as a set of context-free programs, using this application-defined database or cookie as a safe place to store contextual information. Furthermore, these symmetric keys are periodically reissued to ensure their 20<sup>th</sup> freshness and to preclude sufficient time for compromise. In an embodiment of the present invention, the Key Repository process 20 maintains a limited history of old key values so that old data in the application-defined database or cookies will be recoverable. In alternate 25<sup>th</sup> embodiments of the present invention, the database 30 can act as the application-defined database. However, in an embodiment of the present invention, the application-defined database is separate from the database 30.

Context-free programs normally run on what are called context-free servers. It is within the scope of the present invention that the computer system 10 of the present invention can be a context-free server running context-free programs. An example of a context-free program is a web server because the underlying protocol for web transactions, hypertext transfer protocol (http), is stateless and thus free of context. To provide contextual transactions between computer software processes, the embodiments of the present invention can be equipped to retain state

30

(context) information, including confidential information or sensitive information (or data), across one or more instantiations of application processes using the symmetric keys maintained by the Key Repository process. The application in this instance is configured to convey the sensitive context information first by encrypting the sensitive information and then passing the 5 encrypted sensitive information to another instance of the application.

Many transactions involve multiple messages between the submitter and the server. There is often a significant time delay between these messages, especially if the submitter must respond manually, for example, by entering some data at a keyboard. During this delay, however, there is contextual information concerning the transaction that must be saved in order 10 to allow the transaction to continue in an orderly fashion. There are many popular ways to save this information.

One way to save context information is to keep it in the working memory of the server process, and to implement the system such that this same server process instance processes the next incoming message. Unfortunately, this method is expensive, not scalable and not fault-tolerant. The reason this method is expensive is that significant system resources are otherwise unavailable during the waiting period. These significant extra resources must be added to compensate for the processes that are tied up and therefore this method is also not scalable. Moreover, if the process that retains a state fails (or the process in which that process is running fails) the state data is lost and as a result this method is also not fault-tolerant. Accordingly, most 20 modern designs avoid retaining intermediate state in active processes.

A second popular way is to send the contextual information back to the client, and require its re-transmission with the next part of the transaction. Although this method is commonly used by web-based systems using "cookies", it is expensive in terms of communications bandwidth. However, because it does not matter which server instance is invoked, the application is context-free. Unfortunately, if the information includes security information such as secrets, the key that 25 decrypts that data must then be shared among all possible instances of servers which might receive the continuation of the request. The problems associated with the distribution and control of such a key raise the issues addressed in this specification.

A third popular way is to save the contextual information on a database, and to retrieve it 30 when the next part of the transaction takes place. Unlike the well-known cookie method, the database retention method does not increase communications cost, but is still context free.

Server applications written in the second and third methods can be context-free, scalable and fault-tolerant. Within the scope of the present invention, the contextual information will usually include sensitive data (such as cryptographic keys) whose values must be kept private and unmodified. The present invention facilitates the retention of sensitive context by providing 5 authorized application instances with a set of shared symmetric keys, whereby one instance of the server application can use the keys to protect contextual information and another, possibly different application instance, can recover the contextual information because it has access to the same keys. This contextual information can then be transmitted either through the cookie method or in a local database.

10

#### Increasing the Security of the Integrity Key

The Integrity Key 22, being accessible by any operator, is more vulnerable than the Protection Key 24, which requires knowledge of multiple passwords. By linking them together, additional protection of the integrity key is achieved.

As noted above, the Integrity Key 22 protects the integrity of the database 30. The Protection Key 24 protects the confidentiality of sensitive data within the database 30. In addition, the Key Repository process 20 stores within the database 30 a cryptographic link between the two keys, thus making it impossible to modify one without the other.

In the present invention, this cryptographic link is constructed as illustrated in Figure 6 and as described below.

The process begins at step 602. In step 602, a random number source is used to obtain a salt value. Typically, that salt value is 64 bits in length, although salt values of other lengths (longer and shorter) can be used depending upon the desired level of security. In step 606, the Protection Key is obtained. Similarly, in step 608, the Integrity key is obtained. The Integrity 25 Key and the Protection key each consist of the name of the symmetric encryption algorithm, an initialization vector and a key value. In step 610, the Protection Key value, the Salt value, and the Integrity Key value are concatenated together. The concatenated value is then used as input to the hashing algorithm in step 612 to produce a Hash value (i.e. Hash (<Protection-key> | <Salt> | <Integrity Key>) produces the Hash value). The Hash value, the Salt value, and the 30 name of the hashing algorithm are then written to the database in step 614 and the process ends in step 616.

The Integrity Key 22 is revealed by having any operator provide a name and a correct password. Thus, any malicious and sufficiently skilled operator could obtain a copy of the database, use the password to reveal the Integrity Key 22, modify the database 30, and rewrite it. However, since the Protection Key is still safe, none of the private information can be modified without detection, and this includes the shares of the protection key in owner entries. When this corrupted database is returned to the running system and any operator (not necessarily the malicious one) activates the system, the integrity checks in the database will succeed, and the Key Repository process 20 will be unable to differentiate this corrupt database from a legitimate one.

10 Unfortunately for the malicious operator, he did not modify the Protection Key 24. The Key Repository process 20 will detect that the database 30 is invalid (corrupt) at a later time when the owners identify themselves and the original Protection Key 24 is recovered, and the actual key does not produce a valid answer, or the cryptographic link between the keys is tested.

#### Safeguarding Policy Values

One ramification of the arrangement and the application of the present invention is that a change to a policy (numbers, values, etc.) requires the approval of those members of the enterprise that build the master keys. In this way, single individuals cannot grant themselves unauthorized access to the computer system 10.

20 Thus, the present invention contains centralized repository processes for keys, policy, policy decisions, authentication of application programs 40, and authorizations to use enterprise credentials 32. An important security feature of the present invention is the status of Key Repository process 20 as a process. The Key Repository process 20 has a data storage function, in that it stores the Integrity Key 22 and the Protection Key 24 in the working memory of the 25 Key Repository process 20. To protect the security of the master keys within the Key Repository process 20, special functionality on the computer system 10 is utilized to prevent the contents of the Key Repository process from being copied to a swap file or page file. If this lock memory functionality is not used, the Integrity Key 22 and/or the Protection Key 24 could be copied onto the swap file and be compromised. An embodiment of the present invention keeps the Key 30 Repository process 20 in non-swappable protected physical memory.

### Gateways

Alternate embodiments of the present invention are provided with a gateway 60. The gateway 60 provides a separate process that mimics the workstation paradigm for communication with the certification authority 70 per the certification authority's protocol. This 5 enables the system of the present invention to make use of existing libraries that implement the workstation models (paradigms) to communicate with their respective certification authority (in search of the certification authority certificate), without requiring the modification of that library to make it work with the Key Repository process, since those libraries will have access to the enterprise credentials 32. As a result, gateway programs are authorized to the Key Repository 10 process 20 using methods similar to that used to authenticate and authorize application programs 40.

### Summary of Normal Operation Flows

To restate the normal operation of the present invention in terms of the block diagram of Figure 1, an application program 40 attempts to perform some activity on behalf of an enterprise end-entity 31, and requiring the enterprise credentials 32 on the computer system 10. The application 40 asks the Key Repository process 30 if the application 40 is allowed to access the enterprise credentials 32 (that are stored within the database 30). In this sense, the Key Repository process 20 is both a repository and an authenticator. It should be noted that the 20 apparatus and method of the present invention can utilize symmetric as well as the more complex asymmetric key methodologies.

In an embodiment of the present invention, the application 40 asks the Key Repository process 20 for access to an encrypted copy of the enterprise credentials 32. The Key Repository process 20 either provides the password or refuses to provide the password based upon the 25 authorization setup for the enterprise 31 and the application 40. A predefined policy will decide whether or not the Key Repository process 20 provides the password. As mentioned before, the policy framework is stored within the database 30. The enterprise credentials 32, containing the certificates and trust roots of the enterprise 31, are protected by the Integrity Key 22 and the Protection Key 24.

The Integrity Key 22 also protects the policy value that defines the number of Owners needed to change policy values. It also protects the Owner entries needed to re-compute the Protection Key 24, and would detect any tampering with those values.

5 Parameters and Methods

The following is a detailed illustrative discussion of parameters and methods that can be used to implement an embodiment of the present invention. However, it will be recognized by those skilled in the art that considerable modification of this example is possible without departing from the spirit and scope of the present invention.

10

The Keys

All cryptographic systems are based on the concept of keys. For software cryptographic schemes, the longer the key (i.e., the more bits that comprise the key), the more secure the system. Sophisticated cryptography systems utilize multiple keys in schemes that require multiple individuals to be involved with the invocation of system. Multiple keys preclude the compromise of the system by a single individual, thereby increasing the security of the information contained within the system.

The following keying material is used:

20  
21  
22  
23  
24  
25

The Integrity Key

The Integrity Key 22 is a strong symmetric encryption key used to provide integrity for most of the database 30. The method used is a Symmetric Message Authentication Code (MAC) as described by Schneier, and uses any of the symmetric encryption algorithms supported by the Key Repository process 20. The key is generated by the Key Repository process 20 when the database is created, and is regenerated whenever the database is rewritten. The new value of the key is stored in each operator entry, encrypted with that operator's public key, and then saved in the database.

25

Whenever an operator logs on as part of the initial activation of the Key Repository process 20, the operator's password and salt are used to decrypt the private key, which in turn is used to decrypt the Integrity Key 22, which enables the computer program to ascertain that the database has not been tampered with. If the database is already open, this logon provides the

30

Key Repository process 20 with the ability to update the operator's public/private key pair in case one of the parameters which control this operation has changed.

As each subsystem operator is established and the manual password entered, the Key Repository process 20 generates the salt and RSA keys as described above, and the Integrity Key 22 is encrypted with the RSA private key. The reverse step of logging on is as follows:

- 5 a) Use the manually entered subsystem operator's name to find the entry in the database;
- b) Use the password just entered, the Salt value, the hash iteration count, and the algorithm specifications recorded in the database, compute the passkey, which consists of a symmetric key and initialization vector; then, erase the password;
- 10 c) Using the passkey, decrypt the saved private key; then erase the passkey;
- d) Verify that the value just decrypted ends with 8 zero bytes, and then discard those bytes;
- e) Use the private key value to decrypt the Integrity Key 22, and then erase the private key;
- f) If the system is not yet in the starting state, this is the Integrity Key 22 value, otherwise, the value just revealed ought to be the same as the current value of the Integrity Key 22; and
- 15 g) Discard the subsystem operator's password, passkey, retaining only the Integrity Key 22.

#### The Protection Key

The Protection Key 24 is a strong symmetric encryption key used to provide secrecy for sensitive data in the database 30. It is never kept on the database 30 but instead is reconstructed by combining information from the requisite number of Protection Key 24 Owners. This key can be changed upon request.

As each owner is introduced to the Key Repository process 20, the Owner's name and password are used to generate the salt and RSA keys as described above.

25 At a later time, either at database 30 creation, or when the addition of this owner has been approved, a valid Protection Key 24 value will be available. At this point, the Protection Key 24 value is split into N pieces such that any M of them can be used to reconstruct the value, where N is greater than M, and M is greater than one. The methodology is the Bloom-Shamir Secret Splitting algorithm. According to the algorithm, the secret is divided among a plurality of 30 people. As each owner's secret portion is generated, it is encrypted with that owner's public key

and stored in the database 30. Later, when it's time to reconstruct the secret, the needed number of owner's names and passwords are obtained, and:

- a) Use the manually entered owner's name to find the entry in the database;
- b) Use the Integrity Key 22 to verify that the database is okay;
- 5 c) Use the password, the salt value, and the algorithm description to perform a Password Decryption to recover the portion of the shared secret; and
- d) When enough portions have been exposed, use the Bloom-Shamir Secret Splitting algorithm is used to recreate the Protection Key 24.

10 The Trust

There are some assumptions about the kind of attacks from which we have no protection:

- a) An attacker does not have the protection key 24, nor the passwords to retrieve all of them from the database (note that if the attacker has all but one of the required owner's passwords, the last one still protects the protection key);
- b) The attacker will not use supervisory privileges (e.g., root authority) to rummage through the memory image of the Key Repository process 20 and find the keys; and
- 15 c) The attacker will not be able to retrieve the keys from the swap image on the disks.

20 Note, the Operator entry allowed the program to determine that the database 30 had integrity. However, a malicious operator could have generated his own Integrity Key 22, and pass-phrase, and thus could have substituted the files.

The multiple application owner entries allowed the program to decrypt the data.

25 However, this malicious operator could have inserted his own entries as application owners, and created his own master key.

When the Protection Key 24 is finally exposed, the Certification Authority's 70 signing key is decrypted and verified. Remember it must be self-signed. However, this malicious operator could have used his own Certification Authority to generate all the keys.

30 In fact, there is no way the system can differentiate between the "real" database, and a totally fake database that is fully populated. The ultimate protection relies upon dual authentication from the original client program (such as the user at the web browser) and the

internal environment, relying upon trusted copies of trust roots. This situation is identical to that encountered when two principals wish to authenticate each other; both must have independently-authenticated copies of a Root Certificates (or some intermediate trust point) as the root of the trust relationship. However, should any of the normal operators or owners attempt to interact 5 with this fake system, they would immediately detect failures, exhibiting themselves as password failures.

Since the database is protected with both the integrity and privacy checks, no security breach will occur if the database is exposed. However, if the attacker has the passwords of enough owners, all security is moot. It is the responsibility of the customer's practices and 10 procedures to guard against this situation. Prudence therefore dictates that both the passwords and the database should be guarded, making it harder for an attacker to succeed.

### Server Application Programs

Once the Key Repository process 20 has been initiated, and programs authorized to access it, the customer will initiate execution of these programs when needed. The programs will communicate with the Key Repository process 20 to obtain the secrets.

This communication is encapsulated inside libraries that are provided with the present invention, and use a protocol that has the functionality such that the name and content of the program file of the sender can be determined in a trusted manner using operating-system primitives. This precludes the conveyance of this protocol over an open insecure transport 20 medium. The libraries communicate with the Key Repository process 20 using such a protocol. As a result, the Key Repository process 20 can authenticate the application program 40.

It is envisioned that application authentication will be performed when the application first performs credential acquisition (logon), or to restore context, but once done, need not be 25 done again within the instance of this server.

To direct this operation, the libraries must be provided with the name of the Key Repository process 20 in a notation appropriate to the host operating system. If the application is performing a logon function, the name of the enterprise on behalf of whom the application is acting, is also available. The entity name used here is the same name used in the Key Repository 30 process 20 to identify the enterprise credentials 32. How this value is known to the application is application-specific. However, UNIX-like environment variables or command-line parameters

are suitable ways to pass the name of the Key Repository process 20, a method which would enhance portability.

The Key Repository process 20, upon sensing the incoming request, and using functions available from the operating system (such as the name under which it was addressed) 5 understands that the request is originating from an application program 40 and is about to use the agreed-upon protocol to request and to obtain the user's credentials. If the Key Repository process 20 cannot identify the requesting program as being one of the authenticated applications, the incoming request is rejected with a security violation error, and the attempt is logged as a potential security breach. If the application program is successfully authenticated, the Key 10 Repository process 20 ascertains whether this application 40 is authorized to work with the requested credentials. If this test fails, the Key Repository process 20 responds to the requesting application program 40 with an error that is conveyed back to the user's application as a password failure, emulating a logon failure. However, if the test succeeds, the Key Repository process 20 builds a file containing the needed credentials (or uses one already built), encrypts the 15 confidential data with an invented password. This password and the location of the file are returned back to the libraries inside or associated with the application program 40 which completes the acquisition of the credentials by decrypting the credential file. The application 20 program 40 is then informed of a successful logon, and work can commence.

The present invention, therefore, is well adapted to carry out the objects and attain both 20 the ends and the advantages mentioned, as well as other benefits inherent therein. While the present invention has been depicted, described, and is defined by reference to particular embodiments of the invention, such references do not imply a limitation to the invention, and no such limitation is to be inferred. The invention is capable of considerable modification, alternation, alteration, and equivalents in form and/or functions, as will occur to those of 25 ordinary skill in the pertinent arts. The depicted and described embodiments of the invention are exemplary only, and are not exhaustive of the scope of the present invention. Consequently, the present invention is intended to be limited only by the spirit and scope of the appended claims, giving full cognizance to equivalents in all respect.